* البرامج الفرعية :

- عند التعامل مع البرامج الفرعية بلغة C++ هناك برامج فرعية تعيد قيمة وبرامج فرعية لا تعيد قيمة والتي تشبه الإجراء procedure بلغة تربو باسكال لغة C++ تتعامل مع الدوال فقط لذلك نقول انه لدينا دالة تعيد قيمة ويتم ذلك باستخدام الأمر return

return (القيمة العائدة) ;

- هناك دوال لا تعيد قيمة ويتم ذلك باستخدام الكلمة المحجوزة Void
- Void هو أحد أنواع المتغيرات المستخدمة بلغة C++ وتعبر عن اللاشيء ولكن تستخدم مع الدوال التي لا تعيد قيمة وتستخدم ايضا مع المؤشرات .

- تستخدم الدوال من اجل اختصار البرنامج توفير الذاكرة سرعة بالتنفيذ وسهولة كشف الأخطاء

- عند تعريف الدوال بلغة C++ عادة يتم تعريف الدوال في مقدمة البرنامج أما شرح الدوال يتم في نهاية البرنامج الرئيسي مع انه يمكن شرح الدوال في المقدمة
• من اجل تعريف البرامج الفرعية أو الدوال يتم ذلك كما يلي :

1- نحدد نوع المتغيرات للدالة وعند عدم تحديد نوع المتغيرات تكون الدالة من نوع int وإذا كانت الدالة لا تعيد قيمة نحدد نوع المتغيرات Void

2- نضع اسم اختياري للدالة يلي نوع المتغيرات

3- نضيف إلى اسم الدالة قائمة من الوسطاء أو البارامترات إذا وجدت محصورات ضمن قوسين ( ) وعند عدم وجود الوسطاء نترك الاقواس فارغة

4- نكتب الأوامر الخاصة بالدالة محصورة بين قوسي البداية والنهاية علماً يكون الأمر

return (القيمة) ;

هو آخر أمر في الدالة إذا كان المطلوب من الدالة إعادة قيمة اي :

```
{
 :
 :
return (القيمة);
}
```

وصيغتها:

(الوسطاء) اسم الدالة نوع المتغيرات

```
{
    أوامر الدالة
}
```

مثال توضيحي:

البارامترات (أوسطاء)

```
int MM(int a , char c)
{
int m1 , m2 ;   } متغيرات
char c1;
________
________
return (القيمة);
}
```

مثال:

```
void SS(int k)
{
char c;
________
________
}
```

عندئذ يصبح الشكل العام للبرنامج بالشكل التالي:

```
#include <iostream.h>
int MM(int a, char c);
```

```
Void SS(int k);
Void main ( )
{
Long k1, k2;
________
استدعاء الدوال
________
}
int MM(int a, char c)
{
int m1, m2;
char c1;
________
return (القيمة);
}
Void SS(int k)
{
char c;
________
}
```

اثناء تعريف الدالة في مقدمة البرنامج نضع فاصلة منقوطة وعند شرح الدالة لا نضع فاصلة منقوطة

بعد نهاية البرنامج نبدأ شرح الدوال

ملاحظة :

(1) - يجب أن يحصل تطابق وتوافق بين نوع المتغيرات للدالة ونوع القيمة بواسطة الأمر return .

(2) - لا يجوز تعريف دالة بأخرى وانما تعرف الدوال بشكل مستقل ، داخل دالة

(3) - عند التعامل مع الدوال الرمزية (char) يمكن التعامل مع الرمز أو رقم الترتيب ضمن جدول أسكي كود .

البارامترات الفعلية والبارامترات الشكلية .

عند تعريف الدالة نحدد نوع المتغيرات ثم نضعها بعد اسم الدالة وبين قوسين نذكر الوسائط أو البارامترات وهذه البارامترات تسمى بارامترات شكلية .
أما عند استدعاء الدالة من داخل البرنامج الرئيسي يتم ذلك بذكر اسم الدالة تليها قائمة من الوسائط أو البارامترات وهذه البارامترات تسمى بارامترات فعلية .

• مثال :

```
int MM(char c, Long a);    بارامترات شكلية
S = MM('!', 8);            بارامترات فعلية
```

- يجب أن يحصل تطابق وتوافق بين البارامترات الفعلية والبارامترات الشكلية من حيث العدد والنوع .

• مثال :

```
Void MM1(int a1, int a2, int a3);
MM1(2,3,6);
```

المتغيرات المحلية والمتغيرات العامة

المتغيرات المحلية : هي متغيرات تعرف للدالة وتقتصر عمل هذه المتغيرات على الدالة فقط أي لا يجوز استخدامها في دالة أخرى ولا يجوز استخدامها في البرنامج الرئيسي .

• تمرين :
باستخدام مفهوم الدالة اكتب برنامج يجمع عددين صحيحين ويحسب مكعب عدد صحيح .

```
#include <iostream.h>
int sum(int a, int b);
int cb(int k);
Void main()
```

```
{
int  x, y, z, s1, s2;
cout << "\n x = "; cin>>x;
cout << "\n y = "; cin>>y;
cout << "\n z = "; cin>>z;
  s1 = sum(x, y);
  s2 = cb(z);
  cout << "\n s1 = " << s1;
  cout << "\n s2 = " << s2;
}
int  sum(int a, int b)
{
  int c;
   c = a + b;
   return (c);
  }
 int  cb(int k)
 {
  int t;
  t = k * k * k;
  return (t);
  }
```

تمرين :

باستخدام مفهوم الدالة اكتب برنامج يحسب القيمة الصغرى لعددين صحيحين ثم طباعة Borland C++ خمس مرات :

```
#include <iostream.h>
int  min(int a, int b);
void  show();
```

طباعة ٥ أرقام.

```
Void main( )
{
int x,y,m ;
cout << "\n x = "; cin >> x;
cout << "\ny = "; cin >> y;
    m = min (x,y) ;
      Show ( ) ;
cout << "\n m = " << m;
}
int main(int a, int b)
{
int k;
if (a<b)
k=a;
else
k=b;
return(k);
}
Void Show( )
{
int i;
for (i=1 ; i<=5 ; ++i)
cout << "\n Borland C++" ;
}
```

يفضل أن تكون الوسطاء غير المتغيرات

ملاحظة: يمكن للوسطاء في الدالة أن تكون مصفوفة.

مثال:

باستخدام مفهوم الدالة والمصفوفة اكتب برنامج لحساب مجموع عناصر مصفوفة،

```
#includ <iostream.h>
#defin n 5
```

```
int sum(int a[5]);
void main()
{
int b[5], i, s=0;
for(i=0; i<5; ++i)
cin>>b[i];
 s = sum(b);
cout<<"\n s="<<s;
}
int sum(int a[5])
{
int k, j;
```

عادة في الدالة لجمع عناصر المصفوفة.

```
  k=0;
for(j=0; j<5; ++j)
   k=k+a[j];
   return(k);
   }
```

عند التعامل مع الدوال دائماً يتم ادخال المصفوفة في البرنامج الرئيس أما جمع وضرب مصفوفتين يتم في برنامج الدوال

إذا كان الوسيط عبارة عن مصفوفة تذكر اسم المصفوفة من دون الدليل

تمرين وظيفة:

باستخدام مفهوم الدالة والمصفوفة اكتب برنامج (1) لحساب القيمة الصغرى للمصفوفة.

(2) لحساب مصفوفة ضرب عدد صحيح

(3) جمع مصفوفتين ذات بعد واحد.

دوال الإعادة الذاتية:

اثناء كتابة البرنامج يمكن في الواقع استدعاء الدالة لنفسها أكثر من مرة وتستخدم عادة دوال الإعادة الذاتية في العديد من المسائل التطبيقية وعند استخدام دوال الإعادة ذاتية نضع شرط التوقف.

مثال: برنامج حساب n!　　　　5!

```
#include <iostream.h>
int fact (int x);
void main( )
{
int  n, f ;
cin >> n ;
f = fact (n) ;
cout << " \n f = " << f ;
}
int fact (int x)
{
if (x <= 1)
return (1);
else
return (x * fact (x - 1));
}
```

طريقة التطبيق

```
int fact (int 5)
S = fact(5) ;
  = 5 * fact(4)
  = 5 * 4 * fact(3)
  = 5 * 4 * 3 * fact (2)
  = 5 * 4 * 3 * 2 * fact(1)
  = 120
```

حساب a^b

```
if (b == 0)
return (1);        else
a * pow (a, b-1)
```